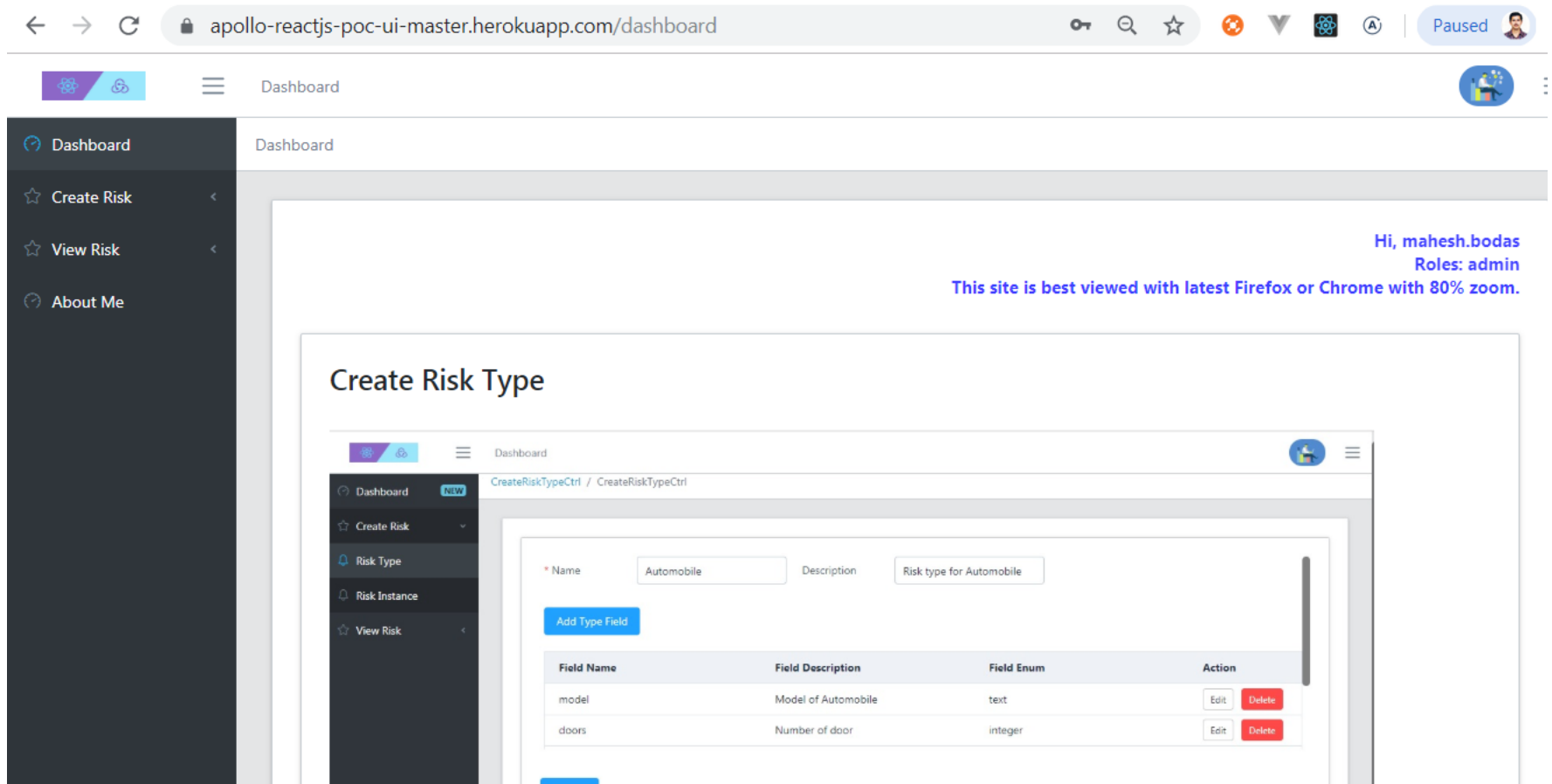


# Home Screen



The screenshot displays the home screen of the Apollo ReactJS POC UI Master application. The browser's address bar shows the URL `apollo-reactjs-poc-ui-master.herokuapp.com/dashboard`. The application's header includes a navigation menu with options like Dashboard, Create Risk, View Risk, and About Me. The main content area features a 'Create Risk Type' form and a table of existing risk types.

Hi, mahesh.bodas  
Roles: admin

This site is best viewed with latest Firefox or Chrome with 80% zoom.

## Create Risk Type

**Name**  **Description**

[Add Type Field](#)

Field Name	Field Description	Field Enum	Action
model	Model of Automobile	text	<a href="#">Edit</a> <a href="#">Delete</a>
doors	Number of door	integer	<a href="#">Edit</a> <a href="#">Delete</a>

# Apollo ReactJS PoC UI

- [ApolloClient and ReactJS PoC application.](#)
- Use above link to access UI.
- Username :- admin (lower case)
- Password :- poctest#1 (lower case)
- Best viewed with Firefox with 80 % zoom
- Above UI refers Web API at following location credentials are same.
- <https://django-poc-server-maheshbodas.herokuapp.com/>

# Salient features

- Developed using ReactJS, ApolloClient, Redux, redux-thunk, ES6, Element React library, Node JS, Serve as a Static server.
- ApolloClient is use to send GraphQL queries and Mutation to GraphQL based Django server.
- Used Redux as global data store for this application. Redux actions and reducers make is easy to shape and store data as needed by UI.

# Salient features (continued)

- Used CoreUI React template so as to reuse Sidebar navigation, Breadcrumbs and common Layout for all pages.
- CoreUI React template itself build using Create React App. CRA is zero configuration template to build, unit test and deploy application on server.
- CRA makes use of Enzyme and Jest for Unit testing. Does not need separate Jest installation or Web pack tweaking. Everything is taken care by CRA.

# Salient features (continued)

- Well thought React components to modularize code.
- Made use of Typescript and React Typescript support to develop generic component that's used in all screen to automate task like showing Page loading Icon, display UI when data is available and show error if any.
- The above component has been reused in all screens of ReactJS PoC application.

# Salient features (continued)

- Role based authorization further limits access to screens and components use for creating Risk Types.
- User can define RiskType (Entity Name) and associated RiskTypeFields (Entity attributes) in one go using header and details table in Create Risk Type screen.
- User can add Risk (Entity Instance Name) and Risk Fields (Entity attribute values) in one go using header and dynamically populated controls in Create Risk screen.

# Salient features (continued)

- Create Risk Instance screen dynamically adds required field validation and field type specific validation for controls related to Risk type fields. (Entity attributes) i.e. Date / float / integer.
- Successful model creation messages are displayed using MessageBox.
- Display various model validation errors using MessageBox

# Salient features (continued)

- View Single Risk display Auto Complete dropdown for selecting Risk Name. Upon selection user clicks Get Risk Detail button to fetch details for given Risk.
- Single Risk Instance data is fetched from GraphQL API.
- Single Risk Instance aggregates Risk Fields as well Field type information. View Single Risk control then dynamically renders appropriate read-only controls based on field type information found in a retrieved Single Risk instance.
- Each dynamically populated Control's value is set with corresponding Risk field (Entity attribute) value

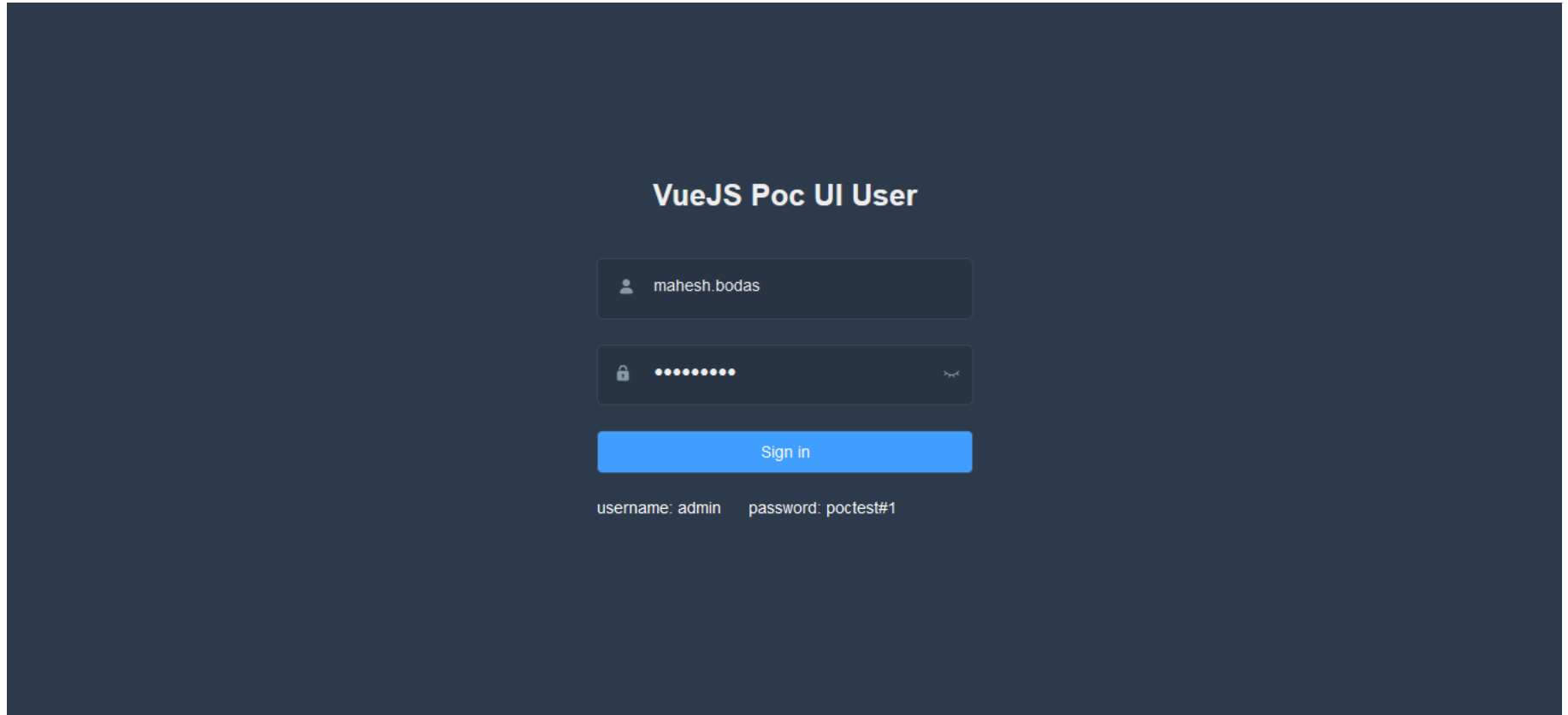




# Salient features (continued)

- View All Risk screen allow user to select RiskType from drop down list.
- Upon RiskType selection, screen fetches Risk instances filtered by selected RiskType and allows user to paginate through list of Risk Instances whose foreign key RiskType matches with the Selected RiskType in Dropdown box.
- GraphQL only support forward pagination. Backward pagination is enabled programmatically by maintaining stack in Redux state store.

# Login Screen

**VueJS Poc UI User**

 mahesh.bodas

 •••••••• 

Sign in

username: admin    password: pocrest#1

# Dashboard showing app highlights

The dashboard features a top navigation bar with a logo, a hamburger menu, the title 'Dashboard', a user profile icon, and another hamburger menu. A left sidebar contains links to 'Dashboard', 'Create Risk', 'View Risk', and 'About Me'. The main content area displays a welcome message: 'Hi, admin' and 'Roles: admin', followed by a note: 'This site is best viewed with latest Firefox or Chrome with 80% zoom.' Below this is a preview of the 'Create Risk Type' form, which includes input fields for 'Name' (containing 'Automobile') and 'Description' (containing 'Risk type for Automobile'), an 'Add Type Field' button, a table of existing fields, and a 'Submit' button. The table has columns for 'Field Name', 'Field Description', 'Field Enum', and 'Action'. The right sidebar lists application features: 'ReactJS PoC', 'Powered by ReactJS ...', 'Development', 'Technology used', 'Unit testing', 'Tools used', and 'Authentication', each with a brief description and a user icon.

Dashboard

Hi, admin  
Roles: admin

This site is best viewed with latest Firefox or Chrome with 80% zoom.

### Create Risk Type

Dashboard / CreateRiskTypeCtrl / CreateRiskTypeCtrl

\* Name: Automobile Description: Risk type for Automobile

Add Type Field

Field Name	Field Description	Field Enum	Action
model	Model of Automobile	test	Edit Delete
doors	Number of door	integer	Edit Delete

Submit

ReactJS PoC

**Powered by ReactJS ...**  
An application to Create and View user defined entities. Implement dynamic forms with validations for entity creation.

Development

**Technology used**  
Developed using ReactJS, ES6, Element UI library, Node JS, Express server. Redux and Redux-Thunk as a middleware.

Unit testing

**Tools used**  
Used Jest, Enzyme and Redux Mock Store for Unit testing of React Components, Redux actions, Redux reducers.

Authentication

**Role based authoriza...**  
Allows only admin user to access all pages in application. Role based authorization

# Create Risk Type screen

- Admin User can define Risk types.
- User Enters Risk type name and description.
- Click Add Risk Type Field button to define Risk type fields (Entity attribute) i.e. Name and Type (String / Date / Integer / Currency).
- Screen allows for editing and deleting Risk Type fields.
- Post Risk type after adding / editing Risk type fields.

# Create Risk Type Image

← → ↻ https://reactjs-poc-ui-master.herokuapp.com/create/creatertypectrl 🔑 🔍 ☆ 🚫 📄 ⚙️ | Paused 👤 ⋮

⚙️ ☁️ Dashboard

🕒 Dashboard  
☆ Create Risk  
🔔 Risk Type  
🔔 Risk Instance  
☆ View Risk  
🕒 About Me

CreateRiskTypeCtrl / CreateRiskTypeCtrl

Field Name	Field Description	Field Enum	Action
houzenumber	House number allotted by corporation	text	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
floors	Number of floors	integer	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
sum	Sum Insurance Amount	currency	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
completion	Construction completion date	date	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

[Hire me](#) © 2019 Mahesh Bodas. Powered by [ReactJS](#)

# Risk Type Data posted to server

```
{"risk_type_name":"Home","risk_type_description":"Risk type for Home","risktype_risktypefields":[{"risk_type_field_name":"houzenumber","risk_type_field_enum":"text","risk_type_field_description":"House Number"}, {"risk_type_field_name":"floors","risk_type_field_enum":"integer","risk_type_field_description":"Number of floors"}, {"risk_type_field_name":"sum","risk_type_field_enum":"currency","risk_type_field_description":"Insurance amount"}, {"risk_type_field_name":"completion","risk_type_field_enum":"date","risk_type_field_description":"Date of completion"}]}
```

# Create RiskType shows Success

The screenshot shows a web browser at the URL `https://reactjs-poc-ui-master.herokuapp.com/create/createrisktypectrl`. The application has a dark sidebar with navigation links: Dashboard, Create Risk, Risk Type, Risk Instance, View Risk, and About Me. The main content area is titled 'CreateRiskTypeCtrl / CreateRiskTypeCtrl' and contains a form with two input fields: 'Name' (with a red asterisk) and 'Description'. Below these fields are two blue buttons: 'Add Type Field' and 'Submit'. A success notification box in the top right corner displays a green checkmark and the text 'Success Risk Type created successfully'. The footer includes the text 'Hire me © 2019 Mahesh Bodas.' and 'Powered by ReactJS'.

Dashboard

CreateRiskTypeCtrl / CreateRiskTypeCtrl

\* Name  Description

Add Type Field

Submit

Success  
Risk Type created successfully

Hire me © 2019 Mahesh Bodas. Powered by ReactJS

# Create Risk Instance screen.

- User can create Risk Instance based on Risk types.
- Select appropriate Risk Type from dropdown box. Screen will fetch Risk type details stored in backend. Dynamically populate appropriate controls based on Entity attributes.
- Fill up all dynamically populated controls that corresponds each Risk Instance field and submit form.
- Only Admin user can define Risk Types.



# Screen shows dynamically populated controls. Ready to accept input values.

← → ↻ 🔒 apollo-reactjs-poc-ui-master.herokuapp.com/create/createriskctrl 🔑 🔍 ☆ 🚫 ▼ ⚙️ ⌛ | Paused 👤

⚙️ 🌐 Dashboard

🕒 Dashboard  
☆ Create Risk  
🔔 Risk Type  
🔔 Risk Instance  
☆ View Risk  
🔔 Single Risk  
🔔 All Risks  
🕒 About Me

CreateRiskTypeCtrl / CreateRiskCtrl

Select Risk Type

* Risk Name	Risk Description
<input type="text" value="risk_name"/>	<input type="text" value="risk_description"/>
* Age	Annual Premium
<input type="text" value="0"/> - <input type="text" value="+"/>	<input type="text" value="\$0.00"/>
* Authorities Contacted	* Auto Make
<input type="text" value="authorities_contacted"/>	<input type="text" value="auto_make"/>
* Auto Model	* Auto Year
<input type="text" value="auto_model"/>	<input type="text" value="0"/> - <input type="text" value="+"/>
* Capital Gain	* Capital Loss

# Screen showing field validation errors.

Browser address bar: <https://reactjs-poc-ui-master.herokuapp.com/create/createriskctrl>

Dashboard

CreateRiskTypeCtrl / CreateRiskCtrl

Select Risk Type:

<p>* Risk Name</p> <input type="text" value="risk_name"/> <p>risk_name is required</p>	<p>Risk Description</p> <input type="text" value="risk_description"/>
<p>Amount</p> <input type="text" value="\$0.00"/>	<p>* Doors</p> <input type="text" value="0"/> <p>doors is required</p>
<p>* Issuedate</p> <input type="text" value="issuedate"/> <p>Issuedate is required</p>	<p>* Model</p> <input type="text" value="model"/> <p>model is required</p>

# Submit when validation succeed

Browser address bar: <https://reactjs-poc-ui-master.herokuapp.com/create/createriskctrl>

Page Title: Dashboard

Navigation Menu:

- Dashboard
- Create Risk
- Risk Type
- Risk Instance
- View Risk
- About Me

Form Fields:

Select Risk Type:

\* Risk Name:

Risk Description:

Amount:

\* Doors:

\* Issuedate:

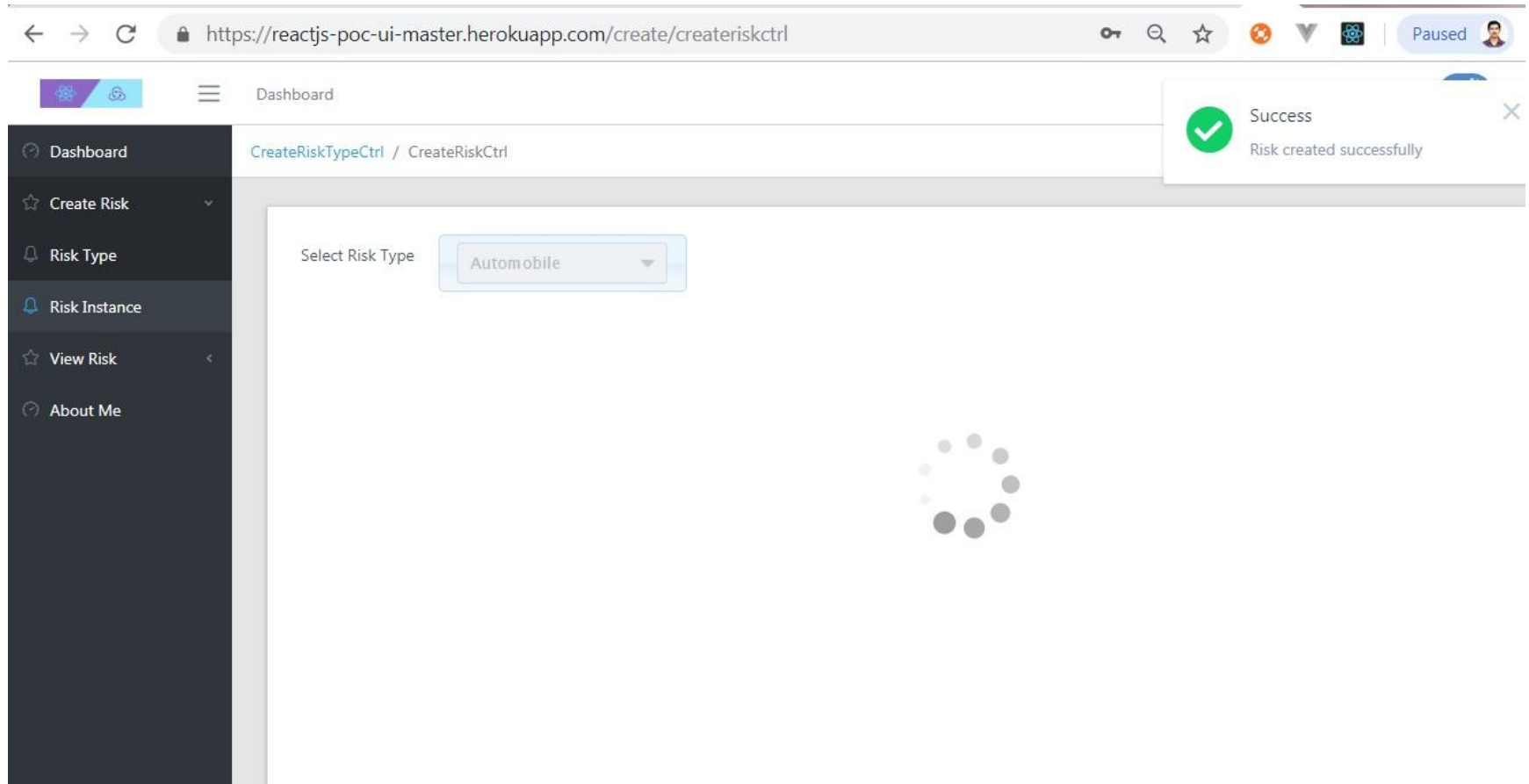
\* Model:

Buttons:

# Posting Risk Instance data to server

```
{"risktype":1,"risk_name":"Mercedes 1","risk_description":"Insurance for Mercedes","risk_riskfields":[{"risktypefield":4,"risk_field_value":"07/14/2016"},{"risktypefield":3,"risk_field_value":30000},{"risktypefield":2,"risk_field_value":4},{"risktypefield":1,"risk_field_value":"Mercedes"}]}
```

# Create Risk Instance showing success



# View Single Risk Instance

- User can view single Risk Instance using this form.
- Select appropriate Risk Name from Auto Complete dropdown box.
- Screen will fetch Risk instance details for selected Risk name. Risk details like name and its attributes values are fetched. Screen will dynamically populate appropriate controls based on Entity attributes type also available in response.
- Each dynamic control's value is set with corresponding entity attribute value.

# View Single Risk Screen

← → ↻ 🔒 apollo-reactjs-poc-ui-master.herokuapp.com/view/viewriskctrl 🔑 🔍 ☆ 🚫 ▼ 🧠 Ⓐ | Paused 👤

⚙️ 🌐 Dashboard

ViewRiskCtrl / ViewRiskCtrl

🕒 Dashboard

☆ Create Risk <

☆ View Risk ▾

🔔 Single Risk

🔔 All Risks

🕒 About Me

Select Risk  Get Risk Detail

**Auto100804**

Auto100804

Auto101421

Auto104594

Auto106186

Auto106873

Auto107181

Auto108270

Auto108844

Risk Name

Risk Description

Age

Annual Premium

Authorities C

Fire

Auto Model

Auto Year

Capital Gain

Capital Loss

AndroidAP  
Internet access

# View All Risks

- User can view Risk Instances using this form
- Select appropriate Risk Type from dropdown box.
- Screen will show one Page of Risk instances  
Risk Name and associated instance fields in  
tabular format for selected Risk Type.
- User can paginate forward and backward  
through Risk records.



# View All Risk Screen

← → ↻ apollo-reactjs-poc-ui-master.herokuapp.com/view/viewriskgrid 🔑 🔍 ☆ 🚫 ▼ 🧠 Ⓐ | Paused 👤

⚙️ ☁️ ☰ Dashboard 👤

🕒 Dashboard  
☆ Create Risk  
☆ View Risk  
🔔 Single Risk  
🔔 All Risks  
🕒 About Me

ViewRiskCtrl / ViewRiskGrid

Select Risk Type

Risk Name	Risk Description	Age	Annual Premium	Authorities Contacted
Auto100804	Auto100804 Risk policy	33	\$ 1,459.96	Ambulance
Auto101421	Auto101421 Risk policy	26	\$ 1,022.46	Other
Auto104594	Auto104594 Risk policy	39	\$ 1,351.10	Fire
Auto106186	Auto106186 Risk policy	25	\$ 1,389.86	Other
Auto106873	Auto106873 Risk policy	37	\$ 894.40	Police

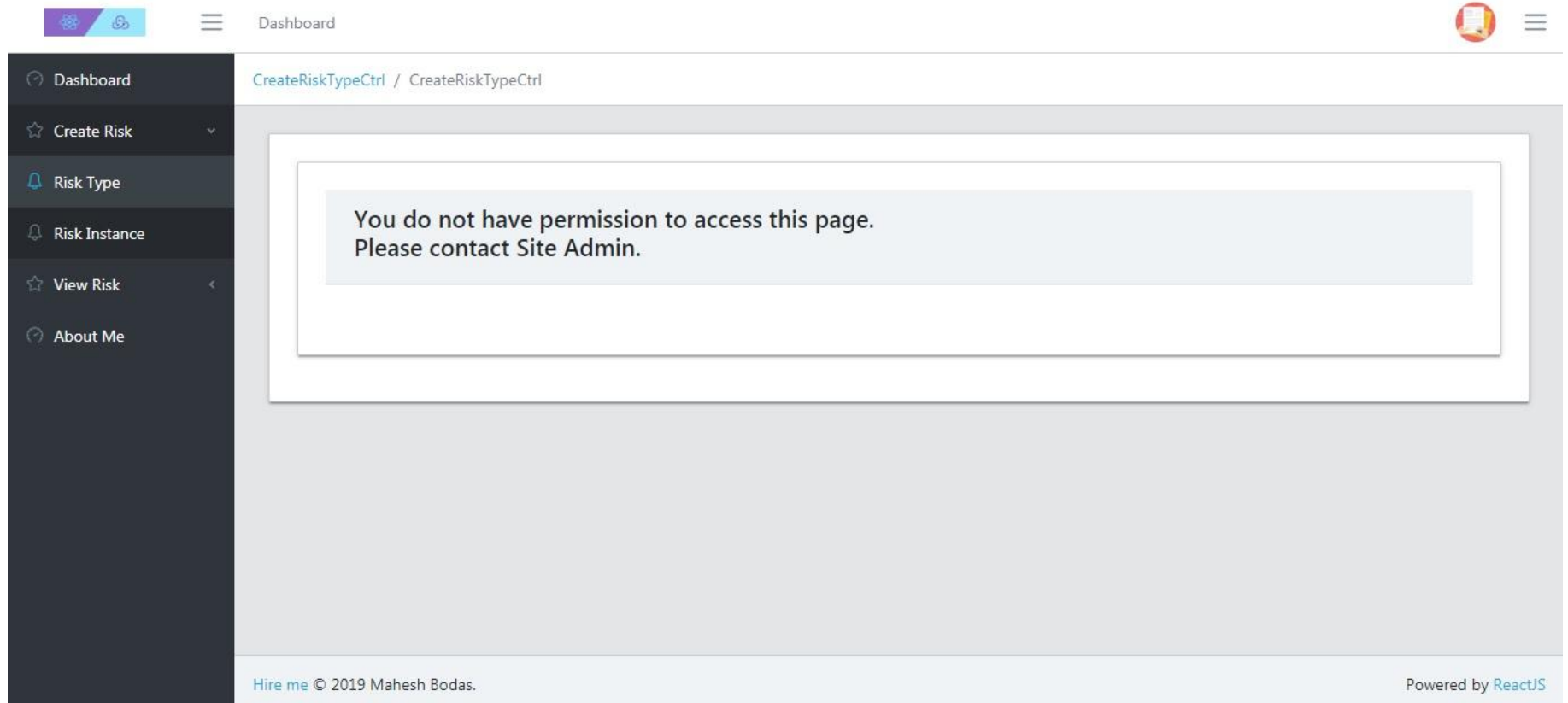
← Previous Page   Next Page →

AndroidAP  
Internet access

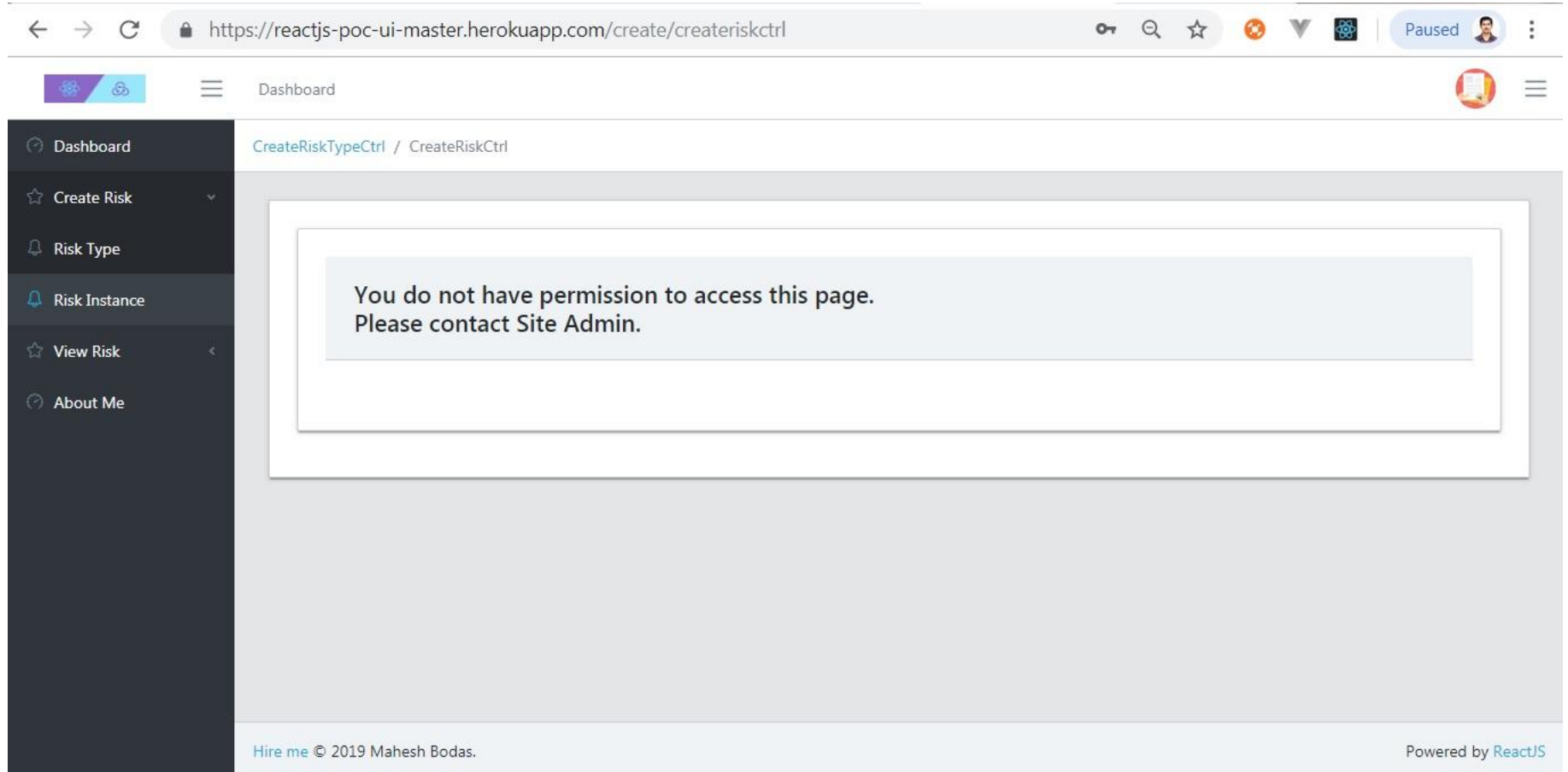
# Non Admin users

- All above slides assumed user logged in has admin privilege.
- Non admin user will be restricted from accessing Create Risk Type and Create Risk Instance page.
- They can view Risk Instances added to system.
- User : editor (lower case)
- Password : test#123 (lower case)

# No access to Create Risk Type



# No access to Create Risk Instance



# Unit testing.

- Used Enzyme, Jest for unit testing React components.
- Used Redux Mock Store for Unit testing of, Redux actions, Redux reducers.

# Known issues.

- Since Django Rest API and UI application are installed using free account on Heroku platform.
- In case if Django Rest API is not invoked in last one hour or more Dyno associated with it goes to sleep.
- As a result it may be possible that when UI invokes API it might timeout.
- In case if you get any error try refreshing page after 2/3 minutes.

# Known issue (continued)

